CSCI 5525 Machine Learning Fall 2019

Lecture 10: Neural Networks (Part 2)

Feb 25th, 2020

Lecturer: Steven Wu *Scribe: Steven Wu*

1 Backpropagation

Now we consider ERM problem of minimizing the following empirical risk function over θ:

$$\hat{\mathcal{R}}(\theta) = \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, F(x_i, \theta))$$

where the ℓ denote the loss function that can be cross-entropy loss or square loss. We will use gradient descent method to optimize this function, even though the loss function is non-convex. First, the graident w.r.t. each W_j is defined as

$$\nabla_{W_j}\hat{\mathcal{R}}(\theta) = \nabla_{W_j}\frac{1}{n}\sum_{i=1}^{n} \ell(y_i, F(x_i, \theta)) = \frac{1}{n}\sum_{i=1}^{n} \nabla_{W_j}\ell(y_i, F(x_i, \theta))$$

We can derive the same equality for the gradient w.r.t. each b_j. It suffices to look at the gradient for each example. We can rewrite the loss for each example as

$$\begin{aligned}\ell(y_i, F(x_i, \theta)) &= \ell(y_i, \sigma_L\left(W_L(\dots W_2\sigma_1(W_1 x_i + b_1) + b_2 \dots) + b_L\right)) \\ &= \tilde{\sigma}_L\left(W_L(\dots W_2\sigma_1(W_1 x_i + b_1) + b_2 \dots) + b_L\right) \\ &\equiv \tilde{F}(x_i, \theta)\end{aligned}$$

where $\tilde{\sigma}_L$ absorbs y_i and ℓ, that is $\tilde{\sigma}_L(a) = \ell(y_i, a)$ for any a. Note that σ'_L can just be viewed as another activation function, so this loss function can just be viewed as a different neural network mapping. Therefore, it suffices to look at the gradient $\nabla_{W_j} F(x, \theta)$ for any neural network F–the gradient computation will be the same.

Backpropagation is a linear time algorithm with runtime $O(V + E)$, where V is the number of nodes and E is the number of edges in the network. It is essentially a message passing protocol.

Univariate case. Let's work out the case where everything is in $\mathbb{R}$. The goal is to compute the derivative of the following function

$$F(\theta) = \sigma_L\left(W_L(\dots W_2\sigma_1(W_1 x + b_1) + b_2 \dots) + b_L\right)$$

For any $1 \le j \le L$, let

$$F_j(\theta) = \sigma_j\left(W_j(\dots W_2\sigma_1(W_1 x + b_1) + b_2 \dots) + b_j\right), \qquad J_j = \sigma'_j(W_j F_{j-1}(\theta) + b_j)$$

All of these quantities can be computed with a *forward pass*. Next, we can apply chain rule and compute derivative with a *backward pass*:

$$
\begin{aligned}
\frac{\partial F_L}{\partial W_L} &= J_L F_{L-1}(\theta) \\
\frac{\partial F_L}{\partial b_L} &= J_L \\
&\dots \\
\frac{\partial F_L}{\partial W_j} &= J_L W_L J_{L-1} W_{L-1} \dots F_{j-1}(\theta) \\
\frac{\partial F_L}{\partial b_j} &= J_L W_L J_{L-1} W_{L-1} \dots J_j
\end{aligned}
$$

Multivariate case. That looks nice and simple. Now as we move to multi-dimensional case, we will need the following multivariate chain rule:

$$
\nabla_W f(Wa) = J^\intercal a^\intercal
$$

where $J \in \mathbb{R}^l \times \mathbb{R}^k$ is the Jacobian matrix of $f\colon \mathbb{R}^k \to \mathbb{R}^l$ at Wa. (Recall that for any function $f(r_1, \dots, r_k) = (y_1, \dots y_l)$, the entry $J_{ij} = \partial y_i / \partial r_j$.) Applying chain rule again:

$$
\begin{aligned}
\frac{\partial F_L}{\partial W_L} &= J_L^\intercal F_{L-1}(\theta)^\intercal \\
\frac{\partial F_L}{\partial b_L} &= J_L^\intercal \\
&\dots \\
\frac{\partial F_L}{\partial W_j} &= (J_L W_L J_{L-1} W_{L-1} \dots J_j)^\intercal F_{j-1}(\theta)^\intercal \\
\frac{\partial F_L}{\partial b_j} &= (J_L W_L J_{L-1} W_{L-1} \dots J_j)^\intercal
\end{aligned}
$$

where J_j is the Jacobian of σ_j at $W_j F_{j-1}(\theta) + b_j$. If σ_j is applying the coordinatewise activation function, then the Jacobian matrix is diagonal.

2 Stochastic Gradient Descent

Recall that the empirical gradient is defined as

$$
\nabla_\theta \hat{\mathcal{R}}(\theta) = \nabla_\theta \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, F(x_i, \theta))
$$

For large n, this can be very expensive to compute. A common practice is to evaluate the gradient on a *mini-batch* $\{(x'_i, y'_i)\}_{i=1}^{b}$ selected uniformly at random. In expectation, the update is moving to the right direction:

$$\mathbf{E}\left[\frac{1}{b}\sum_i \nabla_\theta \ell(y'_i, F(x_i, \theta^t))\right] = \nabla_\theta \hat{\mathcal{R}}(\theta^t)$$

The batch size is another hyperparameter to tune.